Fonctions usuelles en λ-calcul

Léo Gayral

2017-2018

ref : ?

Définition 1 (Entiers de Church)**.** Pour $n \in \mathbb{N}$ on définit l'entier de Church :

$$c_n = \lambda fx.\overbrace{f(f(\dots(fx)))}^{n \text{ fois}} = \lambda fx.f^n x$$

qui correspond *moralement* à n applications successives de la fonction f à la variable x.

On remarque que tout entier c_n est sous forme normale.

Définition 2. On dit que la fonction partielle $\Phi : I \subset \mathbb{N} \to \mathbb{N}$ est λ-calculable si il existe un λ-terme $F \in \Lambda$ tel que :
- Si $n \in I$, alors $Fc_n \to_\beta^* c_{\Phi(n)}$,
- Si $n \notin I$, alors Fc_n n'admet pas de forme normale.

Théorème 1. Les fonctions usuelles sont λ-calculables.

Booléens.
On pose $V = \lambda gd.g$ et $F = \lambda gd.d$. L'intérêt des booléens, ainsi définis, est de naturellement induire un test *si t alors a sinon b* via la formule $\text{if} = \lambda tab.tab$.

On peut en outre encoder le système complet de connecteurs $\{\neg, \wedge\}$ via $\text{not} = \lambda t.tFV$ et $\text{and} = \lambda tu.tuF$.

On peut enfin définir le test d'égalité à 0 via la formule $\mathbb{1}_0 = \lambda n.n(\lambda z.F)V$. □

Somme, Produit.
La fonction arithmétique sur $\mathbb{N}$ la plus élémentaire est le calcul du successeur, ce qui dans notre codage des entiers correspond à appliquer f une fois supplémentaire. Autrement dit, $S = \lambda nfx.f(nf)x$.

Dans une optique similaire, sommer deux entiers n et m revient à composer les applications f^n et f^m. Autrement dit, $+ = \lambda mnfx.mf(nfx)$.

Pour le produit, cependant, il faut composer les *opérateurs* qui élèvent f à une puissance. Autrement dit, $\times = \lambda mnfx.m(nf)x$. □

Couples, Piles.
On peut encoder simplement un couple via la formule $\text{couple} = \lambda xyt.txy$. Étant donnés deux λ-termes $u, v \in \Lambda$, on notera alors $[u, v] = \text{couple}\ uv$ le couple correspondant.

Cette formule est très similaire à if, et on en déduit naturellement un décodage $\pi_1 = \lambda c.Vc$ tel que $\pi_1[u, v] \to_\beta^* V[u, v] \to_\beta^* u$ (resp. $\pi_2 = \lambda c.Vc$).

En outre, en itérant sur des couples de couples, on peut plus généralement représenter une structure de *pile*, en assimilant la pile vide à c_0 par exemple. □

Boucle for*, Prédécesseur.*
Pour calculer le prédécesseur de n, on va procéder par récurrence. On considère la formule $\varphi = \lambda c.\text{couple}\ (\pi_2 c)\,(S\,(\pi_2 c))$, telle que $\varphi\,[c_n, c_m] \to_\beta^* [c_m, c_{m+1}]$. En appliquant $n \in \mathbb{N}^*$ fois φ au couple $[c_0, c_0]$ on obtient $[c_{n-1}, c_n]$.

Moralement, la formule φ représente les opérations effectués au cours d'un pas d'une boucle *for*, qu'on souhaite itérer n fois. On en déduit donc la formule $\text{pred} = \lambda n.\pi_1\,(n\varphi\,[c_0, c_0])$.

On en déduit assez naturellement le calcul de la différence entre n et m – qui doit retourner $\max(n - m, 0)$ en pratique, pour rester dans $\mathbb{N}$ – via la formule $- = \lambda nm.m\ \text{pred}\ n$.

On peut alors définir le minimum entre deux éléments m et n par $\min = \lambda mn.\text{if}\ (\mathbb{1}_0(-nm))\,nm$, et de même $\max = \lambda mn.\text{if}\ (\mathbb{1}_0(-nm))\,mn$.

On peut enfin définir le test d'égalité entre deux entiers via la formule $\mathbb{1}_= = \lambda mn.\text{if}\ (\mathbb{1}_0(-(\max mn)(\min mn)))\,VF$. □

Boucle while*, Calcul de* pgcd.
On va enfin illustrer l'utilisation du combinateur de point fixe pour calculer $n \wedge m$ par l'algorithme d'Euclide. On sait que cet algorithme termine en au plus $\max(m, n)$ itérations, mais on aimerait en pratique interrompre la boucle dès qu'on a effectivement trouvé le pgcd, d'où l'intérêt d'un *while*.

On rappelle que, dans cet algorithme, si $\min(m, n) = 0$, alors $m \wedge n = \max(m, n)$ (initialisation) et sinon $m \wedge n = [\max(m, n) - \min(m, n)] \wedge \min(m, n)$

(récurrence). Dans cette récurrence, notre variant est $\max(m, n)$, qui décroît strictement à chaque pas.

On considère la formule qui encode ce résultat, à savoir :

$$F = \lambda f.\lambda mn.\text{if } (\mathbb{1}_0(\min mn))\,(\max mn)\,(f(-(\max mn)(\min mn))(\min mn))$$

et on pose pgcd le combinateur de point fixe associé, de sorte que $\text{pgcd} \to_\beta^* F\ \text{pgcd}$.

On peut alors montrer, par récurrence sur $\max(m, n)$, que $\text{pgcd}\ c_m c_n \to_\beta^* c_{m\wedge n}$, c'est-à-dire le résultat voulu. □

Remarque 1. Avec des arguments similaires, on peut montrer que toute fonction récursive est λ-calculable, en invoquant des combinateurs de points fixes pour décrire les schémas de récursion primitive et de minimisation.